Texture Synthesis for 3D Shape Representation

Gabriele Gorla[†], Victoria Interrante[‡] and Guillermo Sapiro[†]
[†]*Electrical and Computer Engineering,* [‡]*Computer Science and Engineering*
University of Minnesota

Abstract

Considerable evidence suggests that a viewer's perception of the 3D shape of a polygonally-defined object can be significantly affected (either masked or enhanced) by the presence of a surface texture pattern. However investigations into the specific mechanisms of texture's effect on shape perception are still ongoing and the question of how to design and apply a texture pattern to a surface in order to best facilitate shape perception remains open. Recently, we have suggested that for anisotropic texture patterns, the accuracy of shape judgments may be significantly affected by the orientation of the surface texture pattern anisotropy with respect to the principal directions of curvature over the surface. However it has been difficult, until this time, to conduct controlled studies specifically investigating the effect of texture orientation on shape perception because there has been no simple and reliable method for texturing an arbitrary doubly curved surface with a specified input pattern such that the dominant orientation of the pattern everywhere follows a pre-defined directional vector field over the surface, while seams and projective distortion of the pattern are avoided. In this paper, we present a straightforward and highly efficient method for achieving such a texture and describe how it can potentially be used to enhance shape representation. Specifically, we describe a novel, efficient, automatic algorithm for seamlessly synthesizing, from a sample 2D pattern, a high resolution fitted surface texture in which the dominant orientation of the pattern locally follows a specified vector field over the surface *at a per-pixel level*, and in which seams, projective distortion, and repetition artifacts in the texture pattern are nearly completely avoided. We demonstrate the robustness of our method with a variety of texture swatches applied to standard graphics datasets, and we explain how our method can be used to facilitate research in the perception of shape from texture.

CR categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Texture.
Keywords: Texture synthesis, texture mapping, shape perception, shape representation.

1. Introduction

Adding texture to the surface of a polygonal model can not only profoundly enhance its visual richness, but can also significantly affect our perception of the object's geometry. Certain textures have been shown to impede accurate shape perception, for example by masking faceting artifacts [11]. Others may have the potential to enhance shape perception by emphasizing the lines of curvature of the form [19]. In computer graphics and visualization, where we have the ability to model a textured object in any way that we desire – to both define the texture pattern and define how it is applied over the surface – we have the potential to use texture in highly controlled ways to influence shape perception. Unfortunately, the existing theories on shape perception from texture do not provide sufficient guidance to answer the question of how to best design and apply a texture to a surface in order to facilitate the accurate understanding of its shape.

Researchers in perceptual psychology have been investigating the effects of various texture pattern characteristics on surface shape perception for many years through controlled observer experiments [6, 22, 24, 26, 36]. Unfortunately the scope of these studies has been limited by the capabilities of available rendering and texture-mapping utilities and algorithms. In particular it has not been possible, in general, to map an arbitrary pattern onto an arbitrary doubly curved surface so that the orientation of the pattern everywhere follows a specific predefined vector field at a per-pixel level, while minimizing any distortion of the underlying pattern and avoiding the introduction of pattern discontinuities. Hence most of the studies conducted to date have either not considered the effect of the orientation of the texture pattern with respect to the surface curvature [6, 36], or have been limited to highly restricted synthetic texture

patterns, such as isolated pairs of line segments [26], or have been limited to highly restricted surface geometries, such as the case of singly-curved surfaces [22, 24]. As a result, our ability to gain deeper insight into the specific impact of texture pattern orientation on surface shape perception, and to use this insight to inform theories of shape perception from texture, has been limited. In particular, it is not yet clear to what extent the introduction of texture pattern anisotropy *per se* interferes with surface shape perception [6], or to what extent it is sufficient for unimpeded shape perception to ensure only that the texture pattern does not turn in the surface (i.e. that it does not contain significant geodesic curvature) [22].

In computer graphics and visualization, many feel that the importance to shape perception of the particular alignment over the surface of an anisotropic texture pattern remains open to debate. However we have come to believe, based on informal observations of numerous surfaces under numerous texture conditions, that one's ability to make accurate judgments about the shape of an underlying surface can be significantly influenced both by the characteristics of the texture pattern itself *and* the way in which the pattern is laid down over the surface. Clearly, in order to objectively assess the impact of texture orientation on surface shape perception, it is necessary to conduct further controlled, quantitative experiments. In a recent study using line integral convolution based texture [20], we found indications that observers' shape judgments of a doubly-curved surface are more accurate in the presence of a purely anisotropic texture that follows the first principal directions of curvature over the surface (a special instance of a pattern with zero geodesic curvature) than in the presence of either a purely anisotropic texture that exhibits zero geodesic curvature but does not follow one of the principal directions, or a pattern that contains significant non-zero geodesic curvature (figure 1) [20]. However we found no indications that shape perception is significantly better in the presence of a purely anisotropic principal direction oriented texture pattern than it is in the control case of the purely isotropic texture pattern.

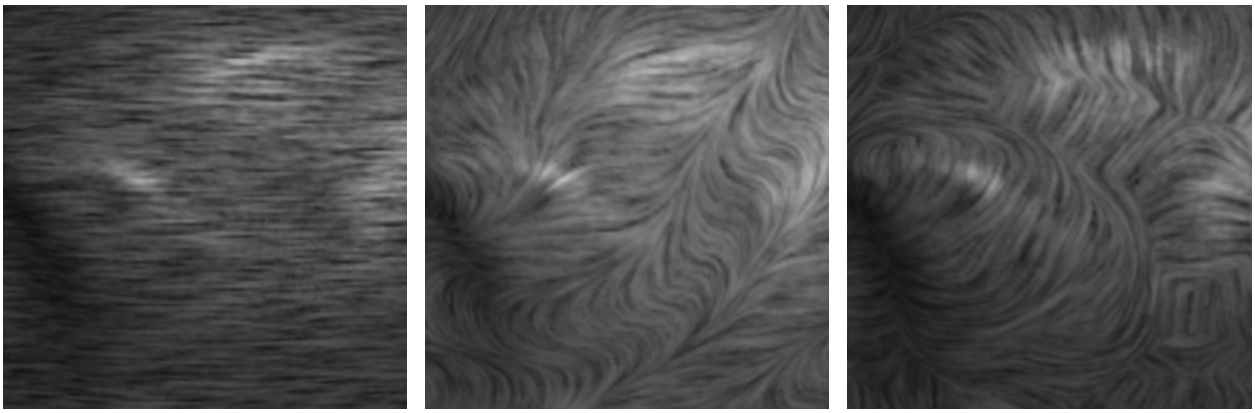

Figure 1: Sample close-up images in an experiment examining the effect on shape perception of differently oriented anisotropic texture patterns synthesized via line integral convolution. Left: a uniformly oriented texture with zero geodesic curvature, Center: a texture with non-zero geodesic curvature; Right: a texture with nearly zero geodesic curvature that follows the first principal direction over the surface. The underlying surface shape is identical in all three cases.

In order to further investigate the key question of how we might best both define and apply a texture pattern to facilitate surface shape perception, we will have to conduct additional studies using a wider variety of surface texture patterns. This requires an algorithm for texturing an arbitrary doubly curved surface with an arbitrary 2D pattern such that the dominant direction of the pattern follows a specified vector field over the surface at a per-pixel level. In this paper we describe such an algorithm that we developed for this purpose. Our algorithm is very straightforward, easy to implement, and highly efficient, and has the potential to be useful for a wide variety of graphics applications that require the aesthetic mapping of a given texture pattern to a given surface, independent of the desire to effectively portray the surface shape.

Given a 2D texture pattern and a polygonal surface model, the historical challenge has been to determine how to apply the pattern to the surface in a manner that minimizies the visual impact of seams and projective distortion while orienting the pattern so that it flows over the shape in a desirable way.

Many different approaches to this basic problem are possible and, concurrently with our work, many similar approaches have been developed. The key distinguishing features of the method that we describe in this paper stem from the fact that it was specifically developed for the purposes of shape representation, in which curvature information is carried by a high resolution texture pattern. Our method has the advantages of being highly efficient for large quantities of texture, very straightforward to implement, and producing high quality results across a wide variety of texture types and models. For the purposes of shape-from-texture investigations, our fitted texture approach is superior to our previous 3D line integral convolution approach [20] because the resulting textured objects can be easily displayed at interactive frame rates using a conventional renderer on a standard PC with texture mapping hardware.

Our technique consists of the following main steps:

- Partition the polygons of the model into contiguous patches, as nearly planar (to prevent distortion) and as nearly similarly sized (to simplify texture map handling) as reasonably possible.
- Compute a vector field over the object, or read a pre-defined field from a file.
- Synthesize the texture pattern over each patch, maintaining pattern continuity across the boundaries with neighboring patches, using an efficient, orientation-adaptive variation of the non-parametric sampling method proposed by Efros and Leung [9].

An example of the results of our algorithm is presented in figure 2.

2. Previous Work

A variety of methods have been previously proposed for texturing polygonal models with patterns that are as free as possible of seams and distortion artifacts.

One method is solid texturing [29, 30, 42], in which the texture pattern is defined over a 3D volume. Particularly good results have been achieved with this method for water, as well as for objects made of wood and stone. However, there are significant challenges in synthesizing 3D textures modeled after sampled materials [17, 8], and current methods for creating custom-fitted 3D textures whose features follow a surface's shape [19] are severely limited in scope and applicability.

Methods for applying 2D image-based texture to arbitrary polygonal models for the most part must balance the inherent trade-off between seams and distortion (one cannot in general apply a 2D image to a non-developable surface without incurring one or the other), employing piecewise flattening in the case of arbitrary parametric [3] or polygonally-defined [25] models, or using careful surface parameterization [23], or pre-distortion of the texture [1, 41] to achieve desired results in other particular cases. Conformal mapping [15] offers a global solution that preserves angles, but not lengths or areas.

Closer to our objectives, Neyret and Cani [28] proposed an excellent technique for achieving seamless and virtually distortion-free mapping of 2D isotropic texture patterns on arbitrary objects via custom-defined triangular texture tiles that are continuous with one another across various of their boundaries. Unfortunately an extension of this method to anisotropic texture patterns is not obvious. Praun et al. [33] subsequently proposed "lapped textures", which provides capabilities that are the most similar to those towards which our method aspires, although the approach that we take is very different. The lapped texture method repeatedly pastes copies of a sample texture swatch onto overlapping patches across a surface after some subtle warping and reorientation to align the pattern with a user-defined vector field. This method produces very good results when used with texture patterns that contain enough high frequency detail and natural irregularity in feature element sizes and relative positions. This is needed to perceptually mask artifacts due to the partial overlap or misalignment of feature elements across patch boundaries. As currently formulated, the lapped texture approach is not particularly well-suited for rigidly structured patterns, such as a checkerboard, or textures such as netting, which are characterized by the global continuity of specific elongated elements. It is also less well-suited for use with vector fields that contain significant high frequency variation. In addition, the lapped texturing process as described in [33] involves considerable amounts of user interaction.

Figure 2: An example of a synthesized surface texture produced by our method. No manual intervention of any kind was employed. This texture was grown from an original 92x92 swatch [5], pre-rotated to 63 orientations each cropped to 64x64 pixels, to cover 291 surface patches at 128x128 resolution following a vector field locally defined by the projection of (0,1,0) onto the tangent plane at each point. The entire process required approximately 12 minutes.

The method that we describe in this paper provides capabilities beyond those offered by these previous methods. It achieves nearly seamless and distortion-free texturing of arbitrary polygonally-defined models with a texture pattern derived from a provided sample. Most importantly, the method is suitable for use with anisotropic patterns. It generally preserves larger scale texture pattern continuity across patch boundaries, does not require manual user intervention, and allows the orientation of the applied pattern to locally follow a specified vector field on a per-pixel basis.

Our method falls into the category of methods that achieve texture pattern continuity without distortion by in effect synthesizing the texture "in place" over the surface of the object. Previous methods in this category include direct painting [16], and reaction-diffusion texture synthesis [37, 41], which yield excellent results for hand-crafted textures and textures modeled after organic processes. Our goal was to achieve similarly good results with automatically synthesized textures that are perceptually equivalent to a given sample swatch.

Our work is perhaps most fundamentally motivated by the impressive advances in texture synthesis methods [17, 7, 32, 44, 9, 39] which have made it possible to create, for an increasingly wide range of patterns, unlimited quantities of a texture that is perceptually equivalent to a small provided sample.

Leveraging research in human texture perception, Heeger and Bergen [17] developed a highly successful method for synthesizing textures that capture the essential perceptual properties of a variety of homogeneous stochastic sample patterns. Their method works by iteratively modifying a random noise image so that its intensity histogram matches the histogram of the sample texture across each of the subbands in a steerable pyramid representation of each image. De Bonet [7] developed a related method based on interchanging elements in the Laplacian pyramid representation of a self-tiling pattern where possible, while preserving the joint-occurrence relationships of features across multiple resolutions. This method yields impressive results for an even wider variety of patterns, though some difficulties remain in preserving larger scale globally significant structure. Several other highly sophisticated texture analysis/synthesis approaches have been subsequently developed [9, 32, 44]. Of these, we chose to follow the texture synthesis approach proposed by Efros and Leung [9] because of its combination of simplicity and quality of output. In this method, a new texture pattern is grown, pixel-by-pixel, by sampling into a provided template pattern and choosing randomly from among the pixels whose neighborhoods are close matches to the yet partially-defined neighborhood of the pixel to be filled in, in the pattern being synthesized.

Subsequent to the appearance of the original Efros and Leung paper, and concurrently with the development of our method, a number of new advancements in 2D and 3D texture synthesis have been achieved. Wei and Levoy [39] proposed a method that addressed one of the most serious concerns with the method of [9] which was speed. Their method used tree-structured vector quantization to improve, by several orders of magnitude, the speed of the search for the pixel with the best matching neighborhood, at the cost of some loss of quality in the resulting synthesized patterns. Ashikmin [2] proposed a method, based on the cutting and pasting of larger areas than a single pixel from the sample texture, that achieved improved results for highly structured textures such as flowers and leaves. Efros and Freeman [10] proposed a new method, called 'texture quilting', that produces even more consistently excellent results across a broad spectrum of texture patterns by specifically maintaining both continuity and coherence across broader local regions of the pattern. Most similar to the objectives of our work, Wei and Levoy [40], Turk [38], and Ying *et al.* [43] all proposed methods for synthesizing a sample 2D texture pattern over an arbitrary mesh in 3D, with the objective of resolving the classical texture mapping problem: to avoid seams and to minimize pattern distortion. The results produced by our method are very similar in many respects to the results produced by these concurrently developed methods, with some subtle differences that will be discussed in a later section. The distinguishing characteristic of our method is that it is optimized for the synthesis of large quantities of *high resolution* texture in which the direction of the texture pattern follows a specified vector field, such as the direction of greatest normal curvature, over the surface at a per *pixel* level, in order to be useful for applications in which one is specifically concerned with the use of texture for facilitating shape representation.

In the remainder of this paper we describe the method that we have developed and the details of its implementation, talk about some of the issues that arise in automatically determining a good way to orient

a texture pattern over a surface, show representative results, and conclude with a discussion of the current limitations of our implementation and directions for future work.

3. Proposed Method

Our proposed method is basically a two step process. First the surface is partitioned into small, almost flat patches. Then, the texture is grown over the planar projection of each individual patch taking into consideration the proper boundary conditions to maintain the continuity of the texture pattern across seams at the patch boundaries. During the synthesis of an anisotropic pattern, the texture is locally constrained into alignment with a specified vector field over the surface. For simplicity, we store the resulting synthesized texture as a collection of separate small images for each patch, although other approaches are certainly possible.

3.1 Partitioning

The goal of the first stage is to partition the mesh into a minimum number of approximately planar patches (collections of triangles). Obviously these are conflicting goals for any closed surface and a suitable tradeoff must be found. To keep our implementation as simple as possible, we restrict patches to be of approximately the same size. Maintaining relatively consistent patch sizes simplifies texture memory management by allowing us to allocate and synthesize texture maps of a consistent fixed resolution for each patch. Please note that mesh partitioning is a task that is common to many computer graphics algorithms and many approaches have been previously described [cf. 34, 27]. In this section we describe for the sake of completeness the details of the particular approach that we used.

Two input parameters define the maximum patch size (which influences the scale at which the texture appears over the surface) and the maximum projective distortion that the user is willing to tolerate. The initial partitioning is done with a greedy algorithm, after which an optimization step is performed to reduce the average projection error.

The process for the initial partitioning can be summarized by the following pseudo code:

while (unassigned_triangles > 0) {
pick an arbitrary unassigned triangle T;
assign T to a new group G;
add to group G all connected triangles C that satisfy:
- Normal(C) • Normal(T) > min_cosine_displacement;
- distance from the center of C to the farthest vertex of T is less than max_dist ;}

The image on the left side of figure 3 shows a representative result after the first stage in the splitting. The green triangles are the reference triangles that define the plane onto which the patch will ultimately be flattened. It is easy to notice that some of the patches obtained at this point are very small and/or contain triangles that are relatively far from being aligned with the reference plane. A refinement pass is used to reduce both the number of patches and the number of triangles that are oriented at a sharp angle to the plane into which they will ultimately be projected. Two simple experimental rules are iteratively applied until no significant improvement is observed:

- remove a patch if it is very small, and its triangles can be added to a neighboring patch without violating the distance constraint;
- reassign a triangle T from patch P1 to a neighboring patch P2 if T borders P2 and is more closely aligned with the reference plane of that patch than with its own.

The image on the right side of figure 3 shows the results after iterative refinement. The simple refinement procedure that we use is not guaranteed to converge to the theoretically optimal result but we have found that the results are consistently good and quite sufficient for our purposes. Since both the splitting and optimization stages are of linear complexity and account, on average, for between 1 and 4% of the total computational time we did not feel the need to improve their speed. The greedy splitting step took about 2 seconds, and the refinement about 10 seconds for the 70,000 triangle data set shown. In the rare event that acceptable results are not achieved in this phase, the splitting process can be repeated using a tighter limit on the acceptable normal error (which will result in more, smaller patches). Increasing the

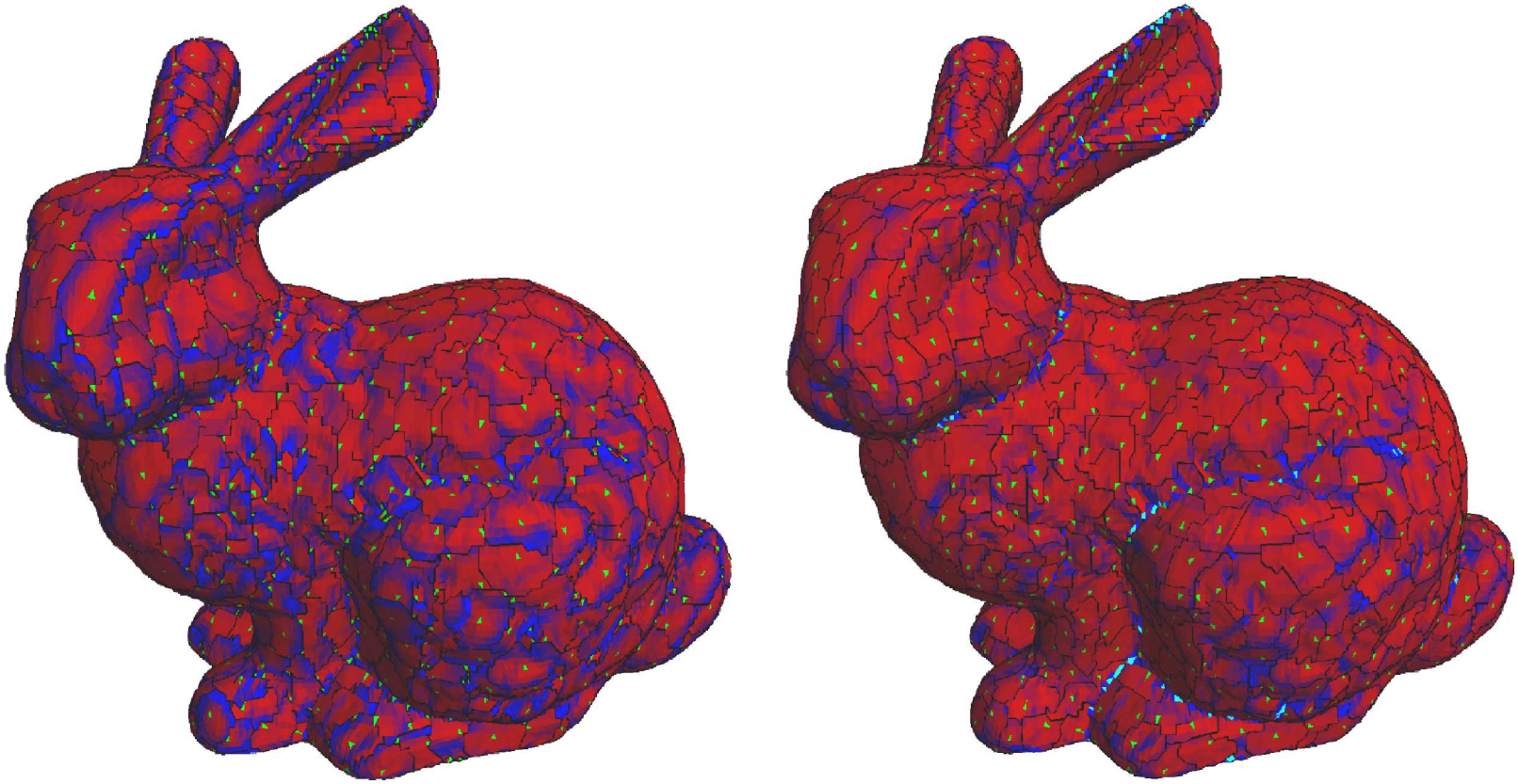

Figure 3: The partitioned bunny, after the first stage of our “greedy” splitting algorithm (left) and after iterative refinement (right), which decreased the number of patches from 988 to 699. Patch boundaries are outlined in black on each model. The green triangles are the seed triangles, which also define the reference plane for each patch. Triangles whose normal directions differ by less than 18 degrees from the direction of the normal to the reference plane, corresponding to a 5% error in the linear projection, are colored red. Triangles up to 25 degrees out of alignment with the reference plane for their patch, corresponding to a 10% error in the projection, are colored blue. Triangles rotated more than 25 degrees from the reference plane (>10% error) are colored cyan.

number of patches does not cause an increase in the computational expense in the subsequent texture synthesis step since the synthesis cost is only dependent on the total number of pixel synthesized. More significant is the issue that with very small patches comes an increased risk that the texture synthesis process will run into difficulties and "grow garbage", either due to the paucity of available contextual information along the shortened boundary, or to the near proximity of mutually incompatible pre-defined boundary conditions.

3.2 Parameterization

After the model has been partitioned into contiguous patches, the triangles comprising each patch are projected onto their common reference plane, and the texture coordinates are defined at each vertex according the coordinates of the projected vertices in the reference plane coordinate system. One major advantage of such a simple parameterization is that there is no need to store the texture coordinates with the output model as they can be easily recomputed at runtime. Adjacent triangles from the neighboring patches, which provide the boundary conditions for maintaining the continuity of the texture pattern during synthesis, are then rotated about their shared edges into the reference plane. We use rotation for these triangles rather than projection to minimize the projective distortion of the texture that we will need to refer to for reference purposes. However, it is necessary to check for the very infrequently encountered cases where it is not possible to rotate each of the adjacent triangles of a particular patch into the projection plane without causing some of these triangles to overlap. This entire process is illustrated in figure 4.

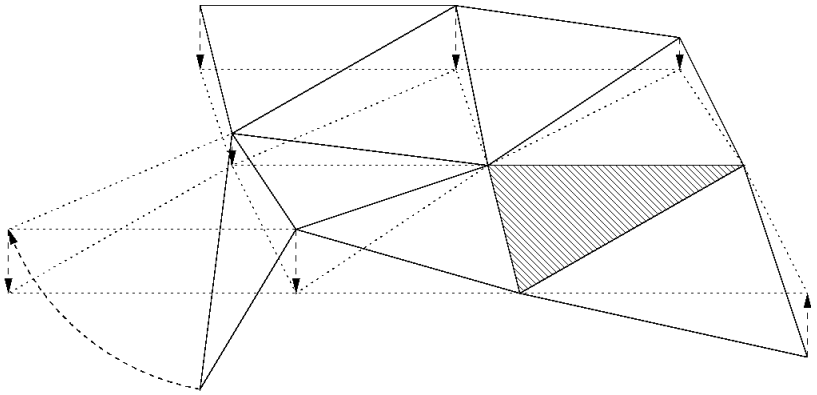

Figure 4: A diagrammatic illustration of the flattening process. The normal of the grey-shaded triangle defines the plane into which the remaining triangles in the patch are projected. The union of these triangles defines the area across which the texture will be synthesized. Neighboring triangles from the adjacent patches are rotated (to minimize projective distortion) into the plane so that any texture already present in these triangles can provide boundary conditions for the texture synthesis, in order to enable the achievement of a seamless final result.

3.3 Synthesis

At this point, we have created a 2D image for each patch containing:

- an area, defined by the projection of the triangles of the patch onto the reference plane, which contains the pixels to be filled by the synthesized texture; and
- an area, defined by the rotation into the reference plane of the neighboring triangles from the adjacent patches, that will hold any previously synthesized texture and provide the boundary conditions necessary to avoid seams due to discontinuities between texture element features in adjacent patches.

It is important to note that the partitioning process described in the immediately previous sections is completely independent from the synthesis algorithm. Any constrained synthesis method that can fill

arbitrary regions with arbitrary boundary conditions could potentially be used, although none of the existing algorithms we reviewed appeared to provide both the image quality and the speed required for this project. The work of Efros and Leung [9] came closest to meeting our needs and thus we elected to follow their general approach, which has been shown to produce good results for a wide range of texture patterns that can be modeled by Markov random fields (i.e. textures whose characteristics are fairly consistent under translation over the image and in which the value at any given point can be fully characterized by the values at its closely neighboring points over a limited range).

In order to make tractable the problem of efficiently synthesizing enough texture to cover a standard model of arbitrary topology at a reasonable resolution, the first objective of our proposed method is to achieve results that are of the same caliber as those demonstrated by Efros and Leung but that require significantly less time, while also preserving the flexible applicability of their approach. To do this, we use a new two-pass search strategy. The first pass, which is exhaustive, is done using a very small unweighted neighborhood (usually between 1/3 to 1/2 of the diameter of the full size neighborhood). The *n* best matches, where *n* is a user-definable parameter, are saved in a list to be processed by the second pass. This two pass approach presumes strong locality in the input textures (which holds true for many natural texture patterns) and has the effect of rapidly eliminating most of the uninteresting part of the search space. The size of this preselect list ultimately determines the overall speed of the synthesis algorithm. We found that some textures produced excellent results with preselect lists of as few elements as the number contained in one scanline of the original image; these we considered easy to synthesize. Others required 4 or even 8 times more elements in the preselect list, and these we considered hard to synthesize. Instances of easy and hard textures are shown in figure 2 and figure 9 respectively. In the second pass, each of the pixels in the preselect list is tested against the full size weighted neighborhood and the error metrics are updated. Among the best 10 or 10% of matches (whichever is greater) a random pixel is chosen and used in the synthesized image. Figure 5 shows a sample result of this synthesis algorithm in the 2D case. The speed of our method does not match the speed of Wei and Levoy's tree-structured vector quantization [39] for the synthesis of rectangular swatches of texture, but unlike their method it does not require the use of a fixed size causal neighborhood, and the results it produces are of consistently high quality. The proposed method is still fast enough to make feasible our goal of growing a fitted surface texture via the Markov random field sampling approach. Figure 6 illustrates the complete texture synthesis process. The image on the upper left represents the state of the model after the synthesis of the first two patches. The image in the upper center identifies the triangles comprising the third patch, with the triangle that defines the plane of the patch rendered in green, and the rest of the triangles rendered in red. The image on the upper right shows the results after the patch is filled with the synthesized texture. The image on the lower left shows the projection of the patch onto the plane. The image in the left center of the lower row shows the texture boundary conditions supplied by the neighboring triangles to this patch. The next two images in the lower row show what the patch looks like midway through and at the end of the texture synthesis step.

3.3.1 Isotropic Textures

As basically formulated, the approach we have just described can be used to cover the surface of a model with an isotropic texture pattern in an orientation-insensitive way. In other words, if we assume that our sample texture pattern is perfectly rotationally symmetric, we can directly use this approach, in its most basic form, to seamlessly synthesize the texture pattern across all of the patches in the model without any special considerations apart from the boundary conditions. However, as we quickly found, there are very few acquired textures that can be used with good results without regard to orientation. Even patterns which we initially believed to be isotropic based on inspection of the 2D sample image revealed unexpected orientation dependencies due to the subtle structuring that stems from the illumination process. Figure 7 illustrates this problem. Notice how the wool texture appears flat when applied without regard to orientation. This is due to the disruption of the pattern of shading.

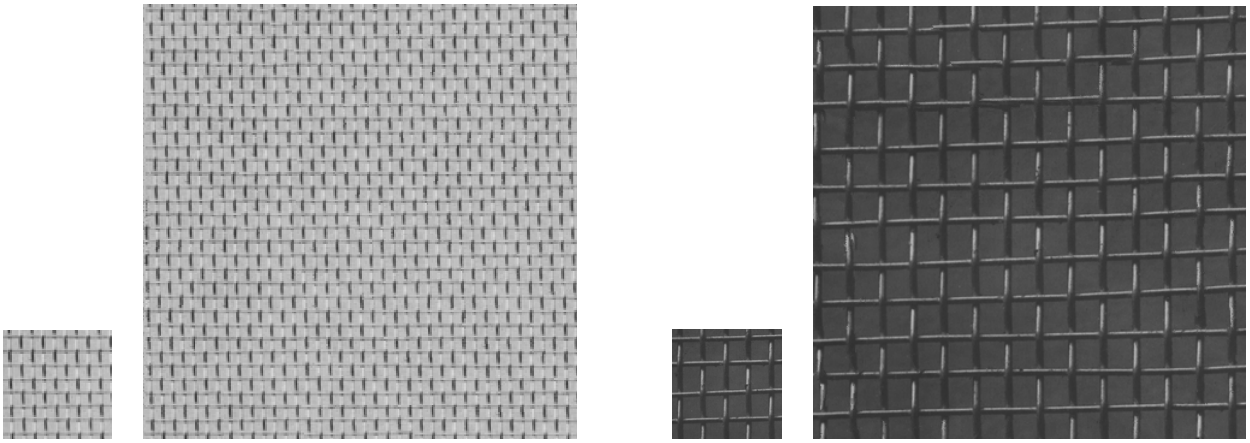

Figure 5: Left: An example of the results of our two-pass texture synthesis method using pattern D06 from the Brodatz album [5]. The speed of the synthesis approach varies from pattern to pattern depending on the sizes of the match-defining neighborhoods and the length of the preselect list. In this case we used a first pass neighborhood of 5x5, a maximum preselect list length of 64, and a second pass neighborhood of 12x12 to synthesize the 256x256 patch on the right from the 64x64 pixel sample on the left in about 73s. Right: an example using D01.

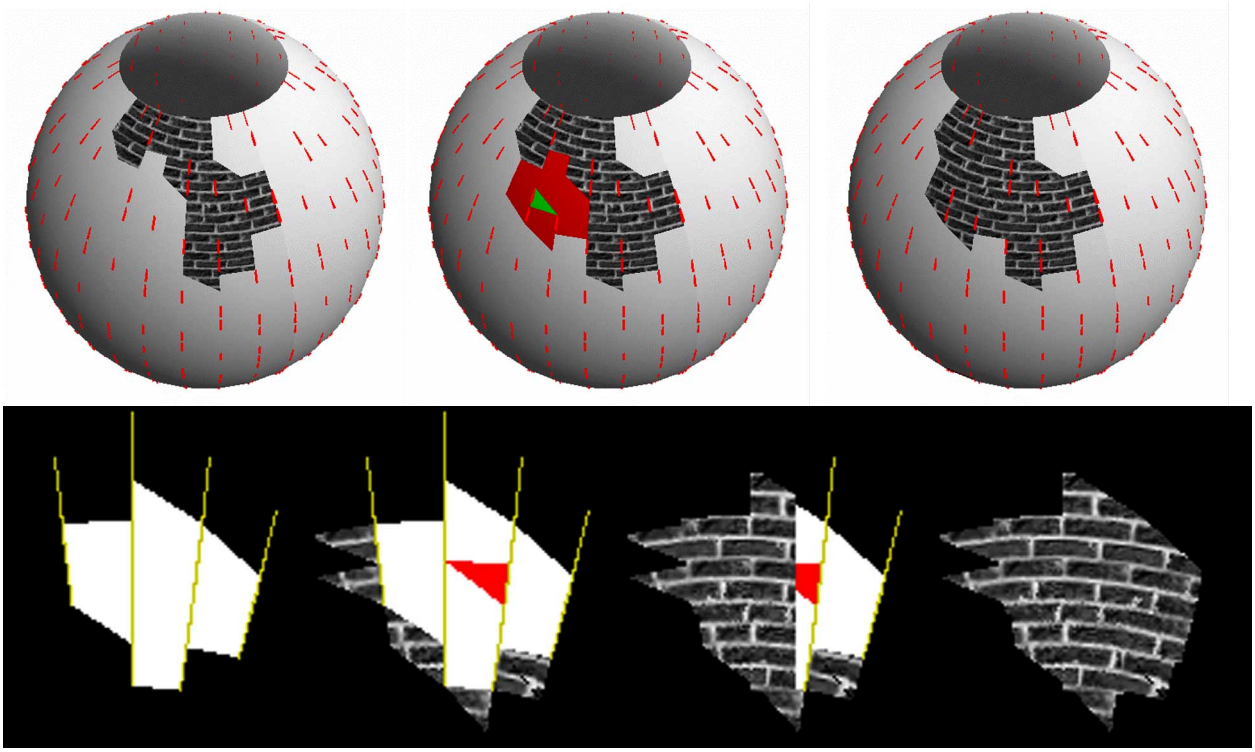

Figure 6: A step-by-step illustration of the basic process of our method. Upper row: the identification of the patch to be synthesized, and the synthesis result. Lower row (from left to right): the planar projection of the patch; the boundary conditions provided by the neighboring textured triangles rotated into the plane of the patch; midway through the texture synthesis process (synthesis is proceeding from left to right); the complete synthesized patch.

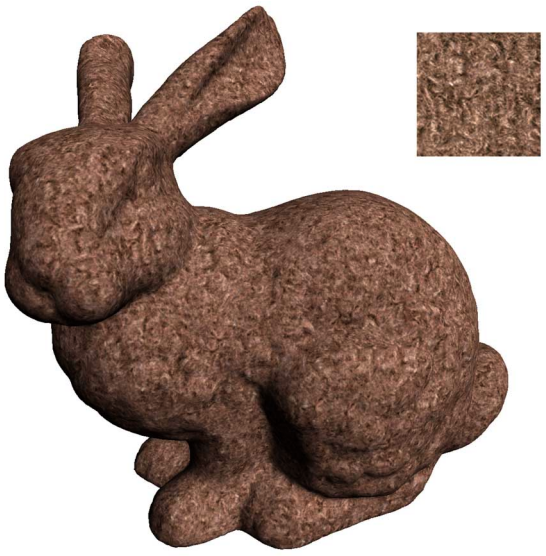

Figure 7: A texture ('wool.bw', from SGI) that originally appeared to be isotropic, reveals its anisotropic nature (due to the effects of shading) when synthesized over the Stanford bunny dataset via an approach in which the texture orientation is allowed to vary arbitrarily between each patch.

3.3.2 Directional Textures

In the vast majority of cases, it is necessary to control the orientation of the texture over the surface. For greatest flexibility, we allow a directional texture to follow any specified direction field. In the next section some examples will be discussed. We note that in the lapped textures method, Praun et al. [33] also align textures on a per patch basis, slightly distorting the parameterization to achieve good local continuity within the patch with the underlying directional specification. In the case of sparse triangulation, they reduce undersampling of the vector field by locally subdividing the mesh.

In our presented method the synthesis algorithm has been enhanced to allow per-pixel texture re-orientation. We pre-rotate the original texture into a quantized number of orientations, and during synthesis perform the search for best-matching neighborhoods in the pre-rotated image that is most closely aligned to the direction locally specified by the vector field. Figure 8 shows a sample of one

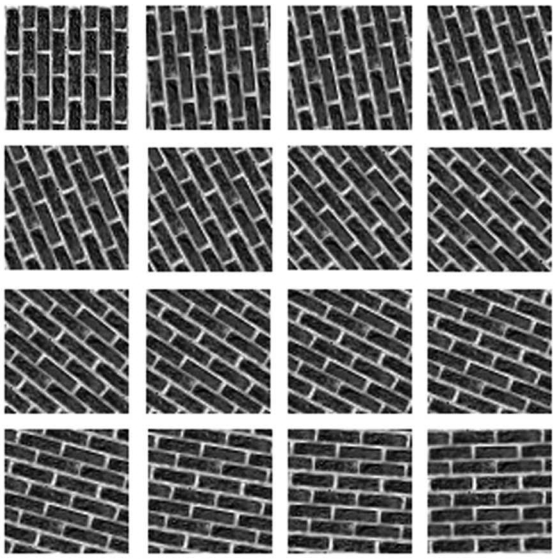

Figure 8: A quadrant of pre-rotated brick texture samples.

quadrant of pre-rotated brick texture. If the number of pre-rotated images is sufficiently high, the synthesized texture will follow the vector field smoothly. For the examples in this paper, we used between 64 to 128 different rotations of the input texture. Although it is of course possible to specify the use of any arbitrary number of pre-rotated images, we did not notice an appreciable increase in the quality of the results when finer quantizations were used.

Searching for matches in pre-rotated texture images allows considerably faster synthesis than would be possible if we had to perform the rotation on-the-fly for each pixel of the texture during synthesis. While it is relatively fast, with modern 3D hardware, to compute any arbitrary rotation of the original image, in most implementations there is a very high cost associated with reading the results from the frame buffer.

3.3.2.1 Constant Direction Fields

We originally began this work with the intent to explore the possibility of applying textures along the principal directions of curvature. Despite the latent potential in that approach, it is not without its difficulties, which we will discuss in greater detail in the following section. We quickly discovered that aesthetic results could be also achieved for a wide range of models using other, much simpler, vector field definitions. Notably, the field of "up" directions, locally projected onto the tangent plane at each point, appears to yield particularly nice results for many textures and datasets, as shown in figures 9 and 10. It is worth mentioning in the context of these two images that we worked hard to challenge our texture synthesis method, testing its performance on difficult texture patterns such as the crocodile skin, which contains potentially problematic sets of features spanning a wide range of spatial frequencies (from 3–21 pixels in diameter), and the square glass blocks pattern, which is a highly structured checkerboard-style design in which irregularities in the size, shape and/or positioning of any of the elements have the potential to stand out especially prominently.

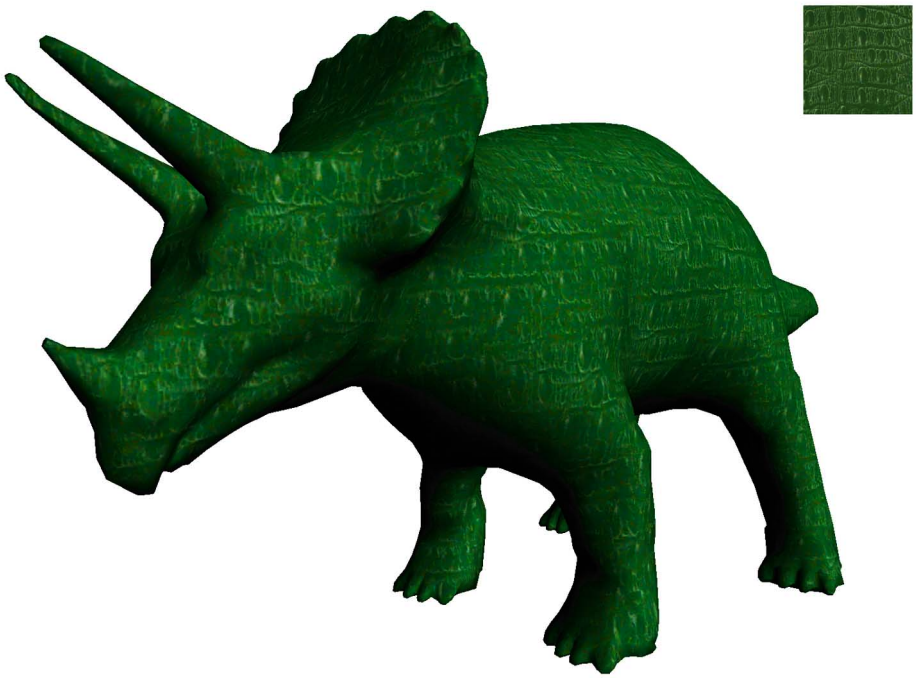

Figure 9: The crocodile skin texture (D10) synthesized over the triceratops model following the direction field (0,1,0) locally projected onto the tangent plane.

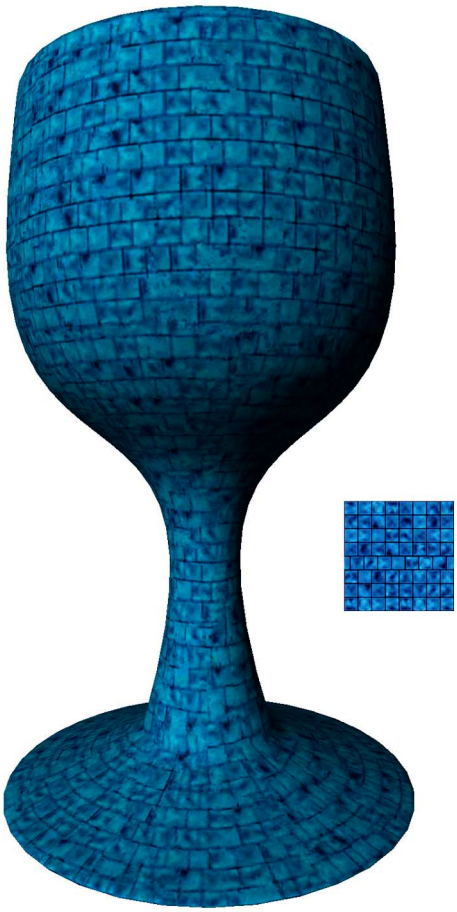

Figure 10: Glass block texture applied to a simple model with a constant directional field. Note the general preservation of continuity in the texture pattern and the relative consistency of the bricks' shapes and sizes, despite scattered artifacts. The direction field is locally given by the projection of the central axis of the object onto the tangent plane of each patch. Some of the patches in this model were resynthesized in a postprocess.

3.3.2.2 Principal Direction Fields

Although the constant direction field produces aesthetic results in many instances, there are also many cases for which it is not well suited. Specifically, it tends to fail for models that do not have a single well-defined intrinsic orientation, and it can not successfully emphasize local shape features. Of greatest intrinsic interest to our ongoing research is the possibility of applying an oriented texture pattern to the surface of an object such that it will be everywhere aligned with the principal directions of curvature.

Recent results in biological vision research support the idea that the principal directions play an important role in surface shape understanding, and we are interested in probing these ideas further through controlled studies of the effects of texture pattern orientation on observers' perception of the 3D shapes of complicated underlying models. Mammassian and Landy [26] have shown that observers' interpretations of line drawings of simple patches are consistent with an inherent bias, among other things, towards interpreting lines on objects as being oriented in the principal directions, supporting an observation made by Stevens [35] nearly 20 years ago. Li and Zaidi [24] examined observers' ability to estimate the relative curvatures of developable surfaces textured with various implicitly or explicitly plaid-like patterns, and concluded that shape perception depends critically upon the observation of changes in oriented energy along lines corresponding to the principal directions. However these ideas remain to be examined in the context of more complicated, arbitrary surfaces, where the first and second

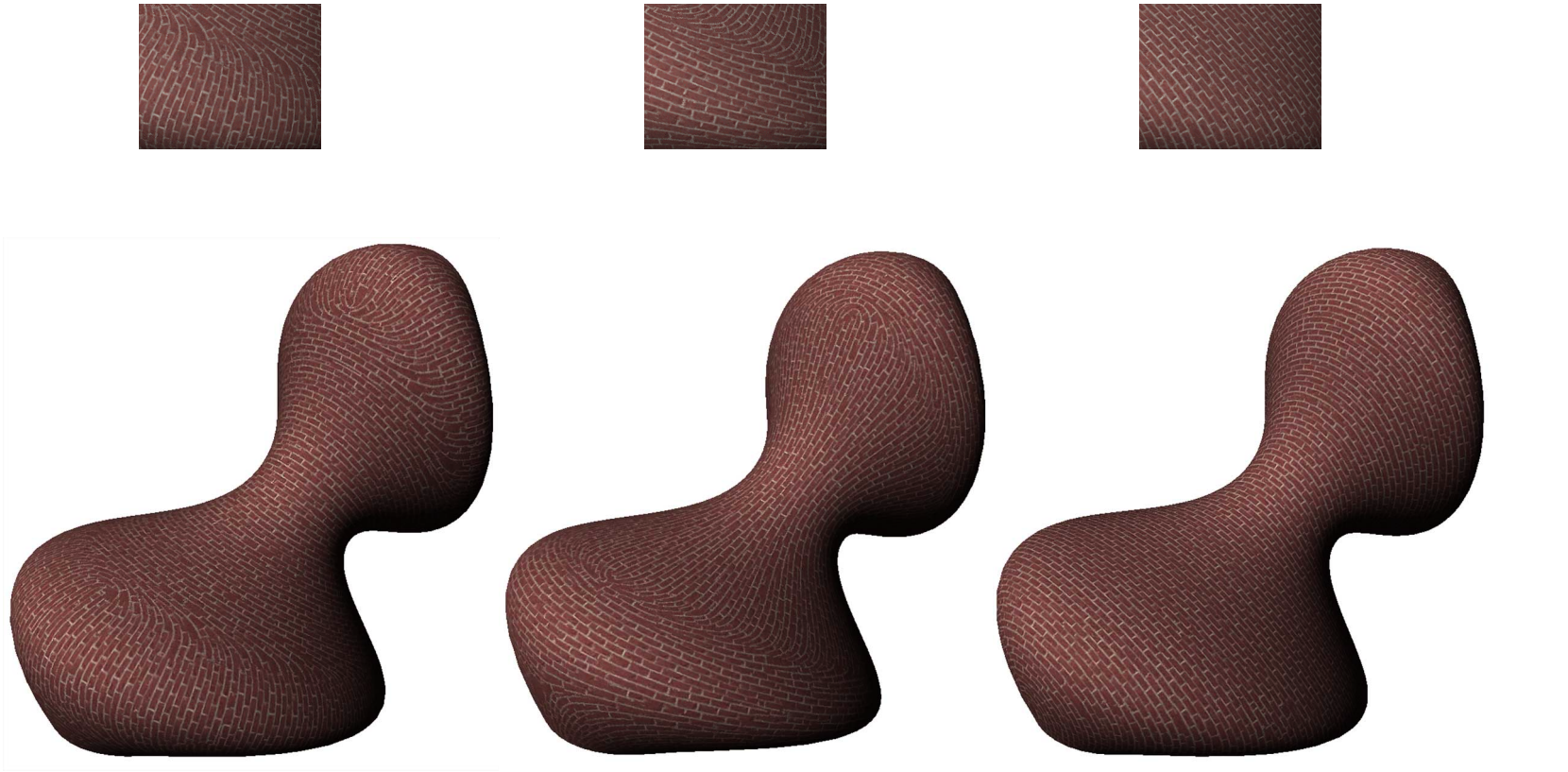

Figure 11: An illustration of the effect that the orientation of a directed pattern over a curved surface can have on our perception of the surface's 3D shape. On the left, the bricks are oriented in the direction of greatest signed normal curvature; in the middle they are oriented in the direction of least signed normal curvature, and on the right they are oriented in the same constant "up" direction used for the models in figures 9-10. Below the entire surface is shown, with the silhouette cues to shape available for reference.

principal directions can switch places numerous times. A significant challenge in this effort is to obtain accurate computations of the principal direction vector fields.

We recently worked with Dr. Jack Goldfeather to develop robust methods for computing smooth, accurate principal direction vector fields across arbitrary polygonally-defined objects [13]. A complementary approach developed by Bertalmio *et al.* [4] has the potential to facilitate the anisotropic smoothing of these fields. Our present results in applying an anisotropic texture over the surface of an object such that its dominant orientation is everywhere aligned with the first and second principal directions are shown in figure 11, and contrasted there with the results obtained using a constant "up" direction. Although it is clearly not possible to prove the benefit of a principal direction oriented texture for shape representation through one or even a handful of representative examples, with the existence of a method for synthesizing a variety of principal direction textures over arbitrary curving forms it becomes feasible to rigorously investigate the impact of various texture orientation strategies on the accuracy of shape perception judgments through controlled observer experiments. We are currently in the midst of carrying out a set of such studies and expect to report the results shortly [21].

4. Implementation

Our system is fully automatic, and does not require user interaction during either the splitting or texture synthesis process. The system has several parameters which can be adjusted by the user to increase the likelihood of obtaining optimal results with different kinds of textures or models. Within the splitting stage, these parameters include: an upper bound on the size of any single patch during splitting (which ultimately affects the scale of the texture on the model) and an upper bound on the angle that the normal of any member triangle can make with the reference direction for a patch (which affects the size and total number of patches). Within the texture synthesis stage, the user may first choose among several possible texture orientation options: to have the texture follow the direction of greatest or least signed or unsigned normal curvature, to follow the projection onto the local tangent plane of a constant specified direction, or to follow no specific direction (in which case the pattern is assumed to be invariant under rotation). With regard to texture synthesis, the user may also control: the sizes of the neighborhoods used in the each pass of the texture synthesis (larger neighborhoods generally increase the computational expense of the synthesis but are sometimes necessary in to preserve features across a range of different scales); the number of first-round preselected locations to be tested for a match on the second pass; and the weighting scheme used over the neighborhood during the matching process.

In most cases, it is sufficient to define the direction of texture synthesis across a patch according to the distribution pattern of previously textured pixels in the boundary region, under the assumption that starting from the side containing the greatest number of previously filled pixels will provide the most stable seed for the synthesis. Unfortunately, this is not always true (see the section on errors below). Certain strongly directional patterns seem to yield better results when the synthesis is performed following the direction of the vector field controlling the texture orientation. This approach was used also by Turk [38]. For quickly varying direction fields, starting the synthesis from an area of the patch in which the direction field is most calm seems to improve the quality of the result.

We use one of two different methods to determine the direction in which the synthesis proceeds from patch to patch. The simple method, which seems to works well on fairly uniform vector fields, is to begin with a randomly chosen patch and proceed to any blank connected patch, filling in any holes at the end. For principal direction vector fields better results can be achieved choosing the blank connected patch in which the difference between the two principal curvatures is greatest. This favors working first in areas over which the principal directions are clearly defined, providing a stable seed for the synthesis of the other patches.

The most computationally expensive part of the algorithm is the texture synthesis, which accounts for 96-99% of the running time. Partitioning and optimizing the patches takes only about 1-3 seconds for simple meshes such as the Venus and triceratops, up to 10-12 seconds for larger meshes such as the 70,000 triangle bunny.

The complexity of the synthesis algorithm is linear with respect to the number of pixels and practically independent from the number of polygons in the input mesh. Growing the weave texture on the Venus model in figure 2 took slightly less than 12 min. The goblet/glass block combination required about 20 minutes. These timings refer to a C++ implementation compiled with gcc[1] running on a standard linux (2.2.16) 933MHz Pentium III PC with a 32Mb GeForce 2 GTS. Table 1 provides a detailed comparison of the speeds of our method and other similar concurrently developed methods. Notice that although some of the other methods report faster execution times, they all involve the synthesis of significantly smaller amounts of texture. The relative efficiency of our method becomes clear when one normalizes for texture quantity.

	Our method	Wei & Levoy [40]	Turk [38]	Ying et al. [43]
Published time for texture synthesis, from a 64x64 sample	10m	82s (TSVQ) 695s (exaustive)	23m	10m (multires.) 3m (coherent)
Texels per vertices synthesized	1,000,000	25,000	256,000	400,000
Machine	P3-933MHz	P2-450MHz	R12K-360MHz	unknown
Estimated machine speed normalization factor	1	2	2.1	unknown
Estimated normalized texture synthesis rate	1600 pixels/s	600 vertices/s (TSVQ) 70 vertices/s (exaustive)	380 vertices/s	unknown

Table 1: A comparison of the speed of our method vs. similar concurrently developed methods.

4.1 Limitations

A major limitation of our method is its inability to capture low frequency texture features across several patches. Since our objective in developing this algorithm was to achieve a method for facilitating shape perception, we assumed that the scale of our desired texture pattern would be substantially finer than the scale of the shape features in our model. Thus our method is not as generic as the methods described in [40], [38] and [43]. However for high resolution textures our method provides substantial speedups over these algorithms.

As described in [8] and [10] all synthesis algorithms based on the "one-pixel-at-a-time" approach are susceptible to "catastrophic failure", in which the algorithm falls into the wrong part of the search space and "grows garbage". Our algorithm is no exception and will occasionally fail and produce undesirable results across all or part of a patch. Depending on the texture, we find that 0-3% of the patches typically contain some synthesis errors. Unfortunately, areas as small as 5x5 pixels (in a 128x128 pixel patch) are easily noticed. The difficulties in automatically detecting such small areas complicate efforts to implement a mechanism for automatic correction. Our current implementation allows the user to interactively select and re-synthesize individual unaesthetic patches after the main automatic synthesis process has finished. Figs 9-11 in this paper show models in which parts of the texture were resynthesized across one or more patches. Figs 2, 5 and 7 show results that were obtained without any such postprocessing

Another limitation that bears mentioning is that we found that the large amounts of texture generated by the synthesis caused problems for certain machine architectures, specifically those that had hard built-in limitations on the amount of texture memory that could be used. When a high level of detail is desired, without any possibility for pattern repetition, the total amount of texture required to cover the mesh can easily exceed the total size of the texture memory. All of the machines that did not allow the storage of

[1] optimization flags -O2 -funroll-loops -fstrict-aliasing -march=i686

textures in main memory failed to display the more complex models (e.g. the crocodile skinned triceratops). Using a texture atlas similar to the one described in [33] might help reduce the texture memory usage, but would incur the cost of incorrect mipmapping and having to store texture coordinates. In our current implementation each patch is stored as a single texture, producing a texture memory waste of up to 25% to 50% depending on the model. However thanks to the OpenGL texture compression extensions, all of the models presented in this paper can maintain interactive frame rates on our standard PC, even in cases where the amount of uncompressed texture exceeds 100Mb.

The method that we have proposed is currently designed to be applied to static models, and we have not thought about how to extend it to the case of deforming animated objects. Modifications to make the texture synthesis process deterministic are an obvious first step toward satisfying this requirement, but it is not immediately clear how one could guarantee that independently synthesized patterns will not differ profoundly from frame to frame as the mesh defining the object is globally deformed.

To achieve good results with the shape-following textures, the direction field must be band-limited: for any given texture there is a maximum spatial frequency that can be followed in the synthesis process and still produce correct results. Nevertheless we found that the method did a good job at singular points on the brick-textured lava lamp object, as can be seen in figure 11. Figures 12a and 12b provide further examples of the behaviour of our method near singularities.

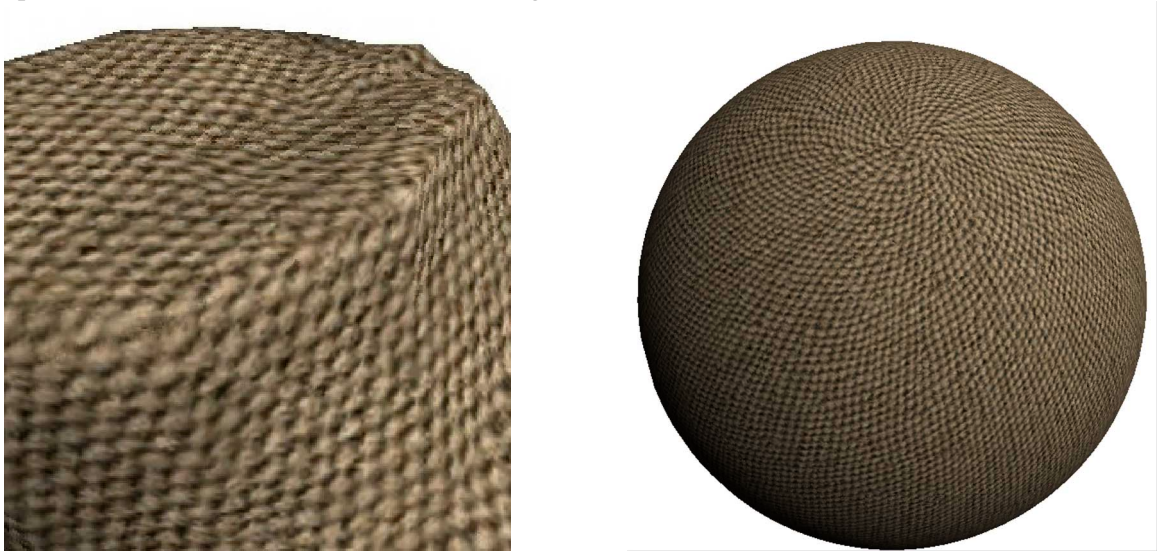

Figure 12: Left: A zoomed-in view of the singular point in the “up” directional field on the Venus model shown in figure 2. Right: A singular point on a sphere.

5. Applications And Future Work

There are many promising applications for this system and many directions for future work. One of the most interesting of these is multi-texturing. On a per-pixel basis it is possible to change not only the direction of the synthesized texture but even the texture itself according to any arbitrary function. Figures 13 and 14 are made using the illumination equation and two and four different textures respectively, each one with 63 rotations. These models, like all of the others in this paper, can be displayed at interactive frames rates on our standard PC.

The multi-texturing methods described in this paper have the potential to be useful for important applications in scientific visualization, for example in encoding a scalar distribution using texture type variations across an arbitrary domain in 2D or 3D. Other direction fields, such as gradient descent, hold promise for different applications, such as non-photorealistic rendering of terrain models (esp. in the case when it is desired to see through the surface). The methods that we have proposed can also be used for the visualization of scientifically computed vector fields over surfaces. An intriguing possible use for an extension of this work is in defining texture mixtures.

Figure 13: A demonstration of multi-texturing, in which the search for matches is performed within an array of different texture types. The texture type index can be defined by any function. In this example, we used the illumination function. In a real application one would want to use something more meaningful, such as soil type over a topographical terrain model.

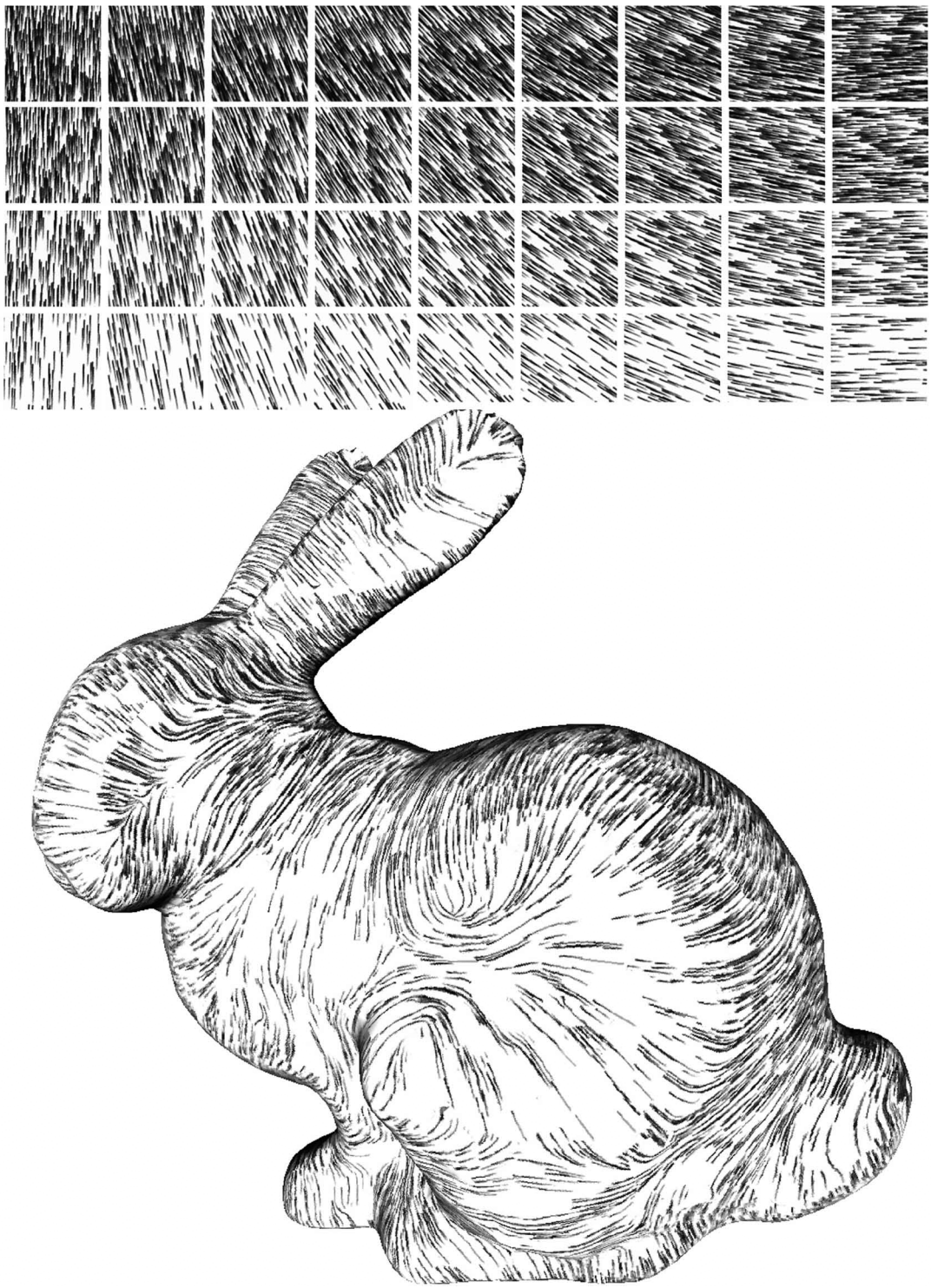

Figure 14: Multiple textures containing lines of different widths applied to an automatically-defined smooth vector field approximating the first principal direction over the Stanford bunny. Indexing along the dimension of varying stroke density was done as a function of the illumination.

6. Conclusions

In this paper we describe a novel surface texturing algorithm with a number of useful applications. Our described method for automatically synthesizing a desired texture pattern in a controlled orientation over the surface of an arbitrary object not only has potential applications in computer graphics, where it provides a simple and efficient solution to the classic problem of fitting a planar pattern to a non-developable surface in a way that minimizes both discontinuity and distortion. More importantly, it also has important potential applications in visualization, where it provides a means for texturing an arbitrary doubly curved surface with an arbitrary anisotropic pattern such that the orientation of the pattern follows a specified directional field over the surface at a per pixel level. With this capability it becomes possible to more thoroughly and directly investigate the effects on shape perception of a broader range of texture characteristics including orientation, and to come closer to answering the question of how best to design and apply a texture that can facilitate the accurate and intuitive perception of a surface's 3D shape.

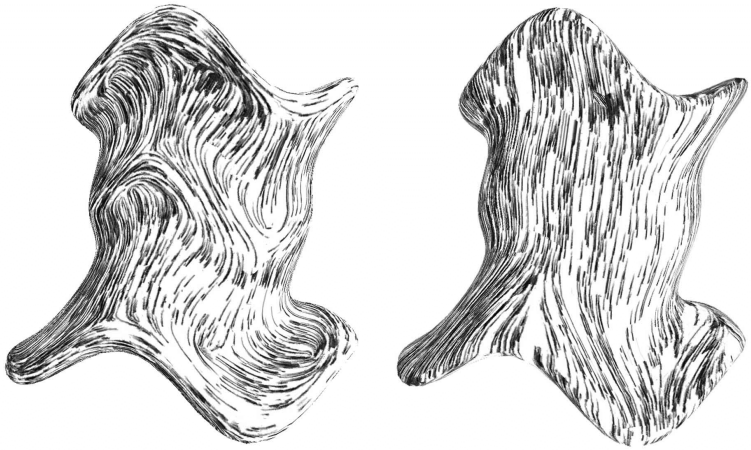

Figure 15: A final demonstration of the effect of texture orientation on shape perception — pen-and-ink style texture synthesized on an unfamiliar surface following the second principal direction (left) vs. a constant 'up' direction (right).

7. Acknowledgments

This work was supported by a University of Minnesota Grant-in-Aid of Research, Artistry and Scholarship. Additional support was provided by an NSF Presidential Early Career Award for Scientists and Engineers (PECASE) 9875368, NSF CAREER award CCR-9873670, ONR-N00014-97-2-0509, an Office of Naval Research Young Investigator Award, Presidential Early Career Award for Scientists and Engineers (PECASE) Navy/N00014-98-1-0631, and the NSF Learning and Intelligent Systems Program.

8. References

[1] Nur Arad and Gershon Elber. Isomeric Texture Mapping for Free-form Surfaces, *Computer Graphics Forum*, *1997*, **16**(5): 247– 256.

[2] Michael Ashikmin. Synthesizing Natural Textures. *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, March 2001, pp. 217 – 226.

[3] Chakib Bennis, Jean-Marc Vézien and Gérard Iglésias. Piecewise Surface Flattening For Non-Distorted Texture Mapping, *Proceedings of SIGGRAPH 91*, pp. 237 – 246.

[4] Marcelo Bertalmio, Guillermo Sapiro, Li-Tien Cheng and Stanley Osher. A Framework for Solving Surface Differential Equations for Computer Graphics Applications, UCLA-CAM report 00-43, December 2000 (www.math.ucla.edu).

[5] Phil Brodatz. Textures: A Photographic Album for Artists and Designers, Dover Publications, 1966.

[6] Jeremy S. De Bonet. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images, *Proceedings of SIGGRAPH 97*, pp. 361 – 368

[7] Bruce G. Cumming, Elizabeth B. Johnston and Andrew J. Parker. Effects of Different Texture Cues on Curved Surfaces Viewed Stereoscopically, *Vision Research*, 1993, **33**(5/6): 827 – 838.

[8] J.-M. Dischler, D. Ghazanfarpour and R. Freydier. Anisotropic Solid Texture Synthesis Using Orthogonal 2D Views, Computer Graphics Forum, **17**(3), September 1998.

[9] Alexei A. Efros and Thomas K. Leung. Texture Synthesis by Non-Parametric Sampling, *Proceedings of the International Conference on Computer Vision*, vol. 2, 1999, pp. 1033 – 1038.

[10] Alexei A. Efros and William T. Freeman. Image Quilting for Texture Synthesis and Transfer, *Proceedings of ACM SIGGRAPH 2001*, pp. 341-346.

[11] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley and Donald P. Greenberg. A Model of Visual Masking for Computer Graphics, *Proceedings of SIGGRAPH 97*, pp. 143 – 152.

[12] Ahna Girshick, Victoria Interrante, Steve Haker and Todd LeMoine. "Line Direction Matters: an Argument for the use of Principal Directions in Line Drawings", *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering*, pp. 43 – 52, June 2000.

[13] Jack Goldfeather. "Understanding Errors in Approximating Principal Direction Vectors", TR01-006, Dept. of Computer Science and Engineering, Univ. of Minnesota, Jan. 2001.

[14] Gabriele Gorla, Victoria Interrante and Guillermo Sapiro. Growing Fitted Textures, *IMA preprint 1748*, Feb. 2001.

[15] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro and Michael Halle. Conformal Surface Parameterization for Texture Mapping, *IEEE Transactions on Visualization and Computer Graphics*, **6**(2), April/June 2000, pp. 181 – 189.

[16] Pat Hanrahan and Paul Haeberli. Direct WYSIWYG Painting and Texturing on 3D Shapes, *Proceedings of SIGGRAPH 90*, pp. 215 – 223.

[17] David J. Heeger and James R. Bergen. Pyramid-Based Texture Analysis/Synthesis, *Proceedings of SIGGRAPH 95*, pp. 229 – 238.

[18] Victoria Interrante, Henry Fuchs and Stephen Pizer. Conveying the 3D Shape of Smoothly Curving Transparent Surfaces via Texture, *IEEE Computer Graphics and Applications*, 1997, **3**(2): 98 – 117.

[19] Victoria Interrante. Illustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution, *Proceedings of SIGGRAPH 97,* pp. 109 – 116.

[20] Victoria Interrante and Sunghee Kim. Investigating the Effect of Texture Orientation on the Perception of 3D Shape, *Human Vision and Electronic Imaging VI*, SPIE **4299**, January 2001, pp. 330 – 339.

[21] Victoria Interrante, Sunghee Kim and Haleh Hagh-Shenas. Conveying 3D Shape with Texture: Recent Advances and Experimental Findings, *Human Vision and Electronic Imaging VII*, SPIE **4662**, January 2002, to appear.

[22] David C. Knill. Contour into Texture: The Information Content of Surface Contours and Texture Flow, *Journal of the Optical Society of America, A*, **18**(1), January 2001, pp. 12-35.

[23] Bruno Lévy and Jean-Laurent Mallet. Non-distorted Texture Mapping for Sheared Triangulated Meshes, *Proceedings of SIGGRAPH 98*, pp. 343 – 352.

[24] Andrea Li and Qasim Zaidi. Perception of Three-Dimensional Shape From Texture is Based on Patterns of Oriented Energy, *Vision Research*, **40**(2), January 2000, pp. 217 – 242.

[25] Jérôme Maillot, Hussein Yahia and Anne Verroust. Interactive texture mapping, *Proceedings of SIGGRAPH 93*, pp. 27 – 34

[26]Pascal Mamassian and Michael S. Landy. Observer Biases in the 3D Interpretation of Line Drawings, *Vision Research*, **38**(18), September 1998, pp. 2817 – 2832.

[27]Alan P. Mangan and Ross T. Whitaker. Partitioning of 3D Surface Meshes using Watershed Segmentation, *IEEE Transactions on Visualization and Computer Graphics,* **5**(4), pp, 308-321.

[28]Fabrice Neyret and Marie-Paul Cani. Pattern-Based Texturing Revisited, *Proceedings of SIGGRAPH 99*, pp. 235 – 242.

[29]Darwyn R. Peachey. Solid texturing of Complex Surfaces, *Proceedings of SIGGRAPH 85*, pp. 279– 286.

[30]Ken Perlin. An Image Synthesizer, *Proceedings of SIGGRAPH 85*, pp. 287 – 296.

[31]Dan Piponi and George Borshukov. Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending, *Proceedings of SIGGRAPH 2000*, pp. 471 – 478.

[32]Javier Portilla and Eero P. Simoncelli. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients, *International Journal of Computer Vision*, **40**(1), October 2000, pp. 49 – 70.

[33]Emil Praun, Adam Finkelstein and Hughes Hoppe. Lapped Textures, *Proceedings of SIGGRAPH 2000*, pp. 465 – 470.

[34]Pedro V. Sander, John Snyder, Steven J. Gortler and Hughes Hoppe. Texture Mapping Progressive Meshes, *Proceedings of ACM SIGGRAPH 2001*, pp. 409-416.

[35]Kent A. Stevens. The Visual Interpretation of Surface Contours, *Artificial Intelligence*, **17**(1-3), August 1981, pp. 47 – 73.

[36]James T. Todd and Robin A. Akerstrom. Perception of Three-Dimensional Form from Patterns of Optical Texture, *Journal of Experimental Psychology; Human Perception and Performance*, **13**(2), 1987, pp. 242 – 255.

[37]Greg Turk. Generating Textures for Arbitrary Surfaces Using Reaction-Diffusion, *Proceedings of SIGGRAPH 91*, pp. 289 – 298.

[38]Greg Turk. Texture Synthesis on Surfaces, *Proceedings of ACM SIGGRAPH 2001*, pp. 347 – 354.

[39]Li-Yi Wei and Marc Levoy. Fast Texture Synthesis Using Tree-structured Vector Quantization, *Proceedings of SIGGRAPH 2000*, pp. 479 – 488.

[40]Li-Yi Wei and Marc Levoy. Texture Synthesis Over Arbitrary Manifold Surfaces, *Proceedings of ACM SIGGRAPH 2001*, pp. 355 – 360.

[41]Andrew Witkin and Michael Kass. Reaction-Diffusion Textures, *Proceedings of SIGGRAPH 91*, pp. 299– 308.

[42]Steven Worley. A Cellular Texture Basis Function, *Proceedings of SIGGRAPH 96*, pp. 291– 294.

[43]Lexing Ying, Aaron Hertzmann, Henning Biermann, and Denis Zorin. Texture and Shape Synthesis on Surfaces, *Proceeedings of the 12th Eurographics Workshop on Rendering*, June 2001.

[44]Song Chun Zhu, Ying Nian Wu and David Mumford. Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling, *International Journal of Computer Vision*, **27**(2), March 1998, pp. 107 – 126.